

### REMARKS

The Applicants respectfully maintain that they have amended the specification and claims to overcome all of the objections and rejections asserted by the Examiner in the Office Action mailed August 28, 2002. Reconsideration and Reexamination is respectfully requested.

### DRAWING OBJECTIONS

In the Office Action, the Examiner asserts an objection to the Figures in paragraph 1 of the Office Action that is related to the item “unitin module 40” and “uninit module 404” found in two places within the specification. In response, the Applicants have amended the specification to conform to the item numbers shown in Fig. 4. Withdrawal of this objection is requested.

### SPECIFICATION OBJECTIONS

In the Office Action, the Examiner asserts an objection to the Specification in paragraph 2 of the Office Action that is related to commonly assigned and co-pending applications identified in the first paragraph of the specification. Specifically, the Examiner objects to the lack of serial numbers for these related applications that were not know at the time of filing. In response, the Applicants have amended the specification to add the requested information.

The Examiner also asserts an objection to the specification because of an inadvertent typographical error found on page 13, line 10. In response, the Applicants have amended the specification to correct the error. Withdrawal of these objections is requested.

### CLAIM OBJECTIONS

In the Office Action, the Examiner asserts an objection to claims 20-23 in paragraph 3 of the Office Action that is related to an inadvertent typographical error in the preamble of claim 20. In response, the Applicants have amended the claim 20 to correct the error. Withdrawal of these objections is requested.

### 35 U.S.C. 112 REJECTIONS

In the Office Action at paragraph 4, the Examiner asserts an rejection under 35 U.S.C. 112, first paragraph, rejecting claims 11-15, 19-23, and 26-31, asserting that they contain subject matter that is not adequately described in the Specification. In response, the Applicants respectfully point the Examiner to Fig. 7 and the corresponding description found within the specification at pages 26-28. The Applicants maintain that this subject matter sufficiently describes the elements recited within the claims. As a result, withdrawal of the rejection is requested.

Additionally, the Applicants respectfully point the Examiner to Fig. 2 and the corresponding description found within the specification. The Applicants specifically maintain that the discussion of pages 11-12 of the specification regarding the networking of computing systems is sufficient to enable one of ordinary skill in the art to make and use the invention recited within claim 31. One skilled in the art knows that programs that are encoded may be transmitted between networked computers and this application as filed is sufficient to enable claim 31.

The Applicants also note that these claims have not been rejected under prior art under 35 U.S.C. 102 or 103.

### 35 U.S.C. 102 REJECTIONS

In the Office Action at paragraph 5, the Examiner asserts an rejection under 35 U.S.C. 102(a), rejecting claims 1-6, 16-17, and 24, asserting that they are anticipated by Guinther et al. (US 6,016,466). Guinther teaches a software performance profiling system. This type of system is similar to the prior art profiling systems expressly described within the background section of the application at pages 1-3. The independent claims 1, 16 and 24 have been amended herein to clarify the differences noted in the application between prior profiling systems in which instrumentation code is inserted for testing and the claimed invention that permanently inserts performance markers into the final product so that the performance testing is performed on a version of an application that is identical to the version that is ultimately considered the delivered product. Guinther teaches the process of instrumenting portions of software that is of interest where testing is to occur. The teachings expressly make a distinction between “instrumented” code and “uninstrumented” code. See Col. 15, lines 21-36 as one example.

In contrast, the claimed invention utilizes performance markers that are permanently inserted into application programs that are used in testing, benchmarking, and profiling the operation of the software. These performance markers are intended to be in the final version of the software that is ultimately delivered to endusers. The Guinther distinction between uninstrumented and instrumented code is consisted with all prior art systems where the instrumentation code is added for the purposes of testing and made to impose little or no impact upon the performance measures created by the instrumentation code. However, the application program is in fact modified to insert the instrumentation code in place of uninstrumented code. In contrast, the claimed invention, as amended, requires the permanent insertion of performance markers into the

application programs that impose little if any overhead to operate, and thus are not removed from the application program when the application has completed its testing. As such, Guinther does not anticipate the claimed invention, as amended, that is recited within independent claims 1, 16, and 24.

Dependent claims 2-6 and 17 recite additional limitations that further distinguish the claimed invention from the teaching found in Guinther and are also allowable for at least the same reasons recited above.

### 35 U.S.C. 103 REJECTIONS

In the Office Action at paragraph 6, the Examiner asserts an rejection under 35 U.S.C. 103, rejecting claims 7-10, 18, and 25, asserting that they are unpatentable over Guinther et al. (US 6,016,466) in light of Levine et al. (US 6,349,406). The Examiner cites Levine for a teaching of subtracting an estimate of an overhead associated with the use of instrumentation code to determine the performance measurements for the actual application program. The claims rejected under 35 U.S.C. 103 all depend from the independent claims discussed above. For the reasons stated above, these claims are patentable over Guinther for at least the reasons stated above. Levine does not remedy this deficiency in Guinther. In fact, Levine teaches away from the permanent insertion of performance markers into an application program. As noted in Col. 8, Levine teaches the use of a JAVA interpreter that is itself instrumented such that performance measures may be obtained. From this, Levine expressly notes that application program itself does not include such markers. Therefore, Levine may not be combined with any teaching for a system in which the use of

permanent performance markers is included within an application program. These claims are now patentable over the prior art of record for at least these reasons.

#### DOUBLE PATENTING REJECTION

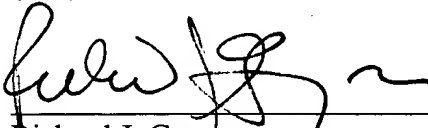
In paragraph 7 of the Office action, the Examiner asserts a provisional double patenting rejection for the claims in this application with the related application, Serial No. 09/606,925. This application is commonly assigned and the Applicants will submit a terminal disclaimer in this application upon receipt of an indication that the applications are otherwise allowable. If the Examiner wishes to request these prior to the entry of the next Office Action because the applications are deemed otherwise allowable, the Applicants respectfully request the Examiner contact the undersigned counsel and such terminal disclaimers will be provided to place both applications in condition for allowance.

#### CONCLUSION

For all of the above reasons, the Applicants respectfully maintain that the pending claims, as amended, are now patentable over the prior art of record. For these reasons, the Applicants respectfully request that the above objections and rejections be withdrawn and the application be passed for allowance.

Respectfully submitted,  
MERCHANT & GOULD P.C.  
P.O. Box 2903  
Minneapolis, Minnesota 55402-0903  
(612) 332-5300

Date: 13 Dec 2002

  
Richard J. Gregson  
Reg. No. 41,804

**VERSION WITH MARKINGS TO SHOW CHANGES MADE**

**In the Specification**

Please amend the specification to replace the paragraphs indicated below with following text. No New Matter is being added in this amendment.

Please amend the paragraph found on page 1, lines 5-10 as follows:

The instant application is related to two concurrently assigned and concurrently filed U.S. Patent Applications, PERFORMANCE MARKERS TO MEASURE PERFORMANCE OF FEATURES IN A PROGRAM, Serial No. [ ] 09/606,961, filed [ ] June 29, 2000, (Attorney Docket M&G 40062.69-US-01) AND PERFORMANCE MARKERS TO MEASURE BENCHMARK TIMING OF FEATURES IN A PROGRAM, Serial No. [ ] 09/606,925, filed [ ] June 29, 2000, (Attorney Docket No. 40062.69-US-02).

Please also amend the paragraph found at page 13, lines 7-18 as follows:

Fig. 3 illustrates another application program testing system according to another embodiment of the present invention. An application program 101 being tested consists of an application main body module 301, one or more application specifically defined linked libraries 302-304, one more application specifically defined dynamically [link] linked libraries 312-314, a performance benchmark statically linked library 305, and a performance benchmark dynamically linked library 315. In the past, application programs typically consisted of all the above modules

with the exception of the performance benchmark statically linked library 305 and the performance benchmark dynamically linked library 315. An application developer who writes the application program 101 has specified a collection of processing modules, which may not be organized into the various libraries. These modules are combined together to create the application program 101 that the application developer wishes to test.

Please also amend the paragraph found at page 17, line 15 through page 18, line 2 as follows:

Once all the processing has been completed, the unInit module [404] 403 is called using unInit call 413. This module [404] 403 also checks to determine if the Init module 401 has indicated that benchmark processing is to occur. If this benchmark processing is not to occur, the linked unInit module [404] 403 simply returns to the application program 101. If the benchmark processing is to occur, the linked uninit module [404] 403 calls the corresponding unInit DLL module 423. The unInit DLL 423 module retrieves all of the benchmark data records from the memory block 424, formats them appropriately, and stores these records within the mass storage device 111. Once these records reach the mass storage device 111, further analysis, review, and subsequent processing may be performed by the application developer as needed.

### **In the Claims**

Please amend claims 1, 16, 20 and 24 as shown below:

1. (ONCE AMENDED) A computing system having a mass storage device and a system timer for obtaining benchmark timing for a portion of an application program execution, the

application program having permanently inserted performance markers, the computing system comprising:

a mass storage system;

an init module for determining if the timestamp data is to be collected during the operation of the application program;

a performance marker module for obtaining and storing the timestamp data for later retrieval;

an uninit module for formatting and storing the obtained timestamp data into a data file within the mass storage device that permits retrieval after the termination of the application program; and

a performance benchmark data post processing module for determining the benchmark timing from two or more timestamp data entries;

wherein the performance marker module is executed at predefined points corresponding to the permanently inserted performance markers within a plurality of processing modules within the application program.

16. (ONCE AMENDED) A method for obtaining benchmark timing for a portion of an application program execution, the application program having permanently inserted performance markers, the method comprising:

permanently inserting one or more [code] performance markers into the application program at predefined locations within the application program corresponding to the point at which benchmark timing data is desired;

determining if benchmark timing data is to be collected at each [code] performance marker by checking for the existence of processing modules identified by an identification key within a system registry;

if benchmark timing data is to be collected at each [code] performance marker:

generating a benchmark data record containing the collected benchmark timing data each time the [code] performance markers are reached;

storing the benchmark data records within a data memory block within the processing modules identified by the identification key within the system registry;

retrieving the benchmark data records from the data memory block for transfer to first data record in a Raw Data Table device once all of the run-time internal state data has been collected; and

processing the first data records stored within the Raw Data Table to generate second data records in a Processed Data Table that estimate the benchmark timing defined between two benchmark data records.

20. (ONCE AMENDED) The method [system] according to claim 19, wherein the second data record of the Processed Data Field comprises a ResultID field, a Reboot Iteration field, a Launch Iteration field, a Marker Iteration field, a Marker Pair ID field, and a Seconds Field.

24. (ONCE AMENDED) A computer data product readable by a computing system and encoding a computer program of instructions for executing a computer process for obtaining run-time internal state data within an application program, the application program having permanently inserted performance markers, said computer process comprising:

permanently inserting one or more [code] performance markers into the application program at predefined locations within the application program corresponding to the point at which benchmark timing data is desired;

determining if benchmark timing data is to be collected at each [code] performance marker by checking for a processing modules identified by an identification key within a system registry;

if benchmark timing data is to be collected at each [code] performance marker:

generating a benchmark data record containing the collected benchmark timing data each time the code markers are reached;

storing the benchmark data records within a data memory block within the processing modules identified by the identification key within the system registry;

retrieving the benchmark data records from the data memory block for transfer to first data record in a Raw Data Table device once all of the run-time internal state data has been collected; and

processing the first data records stored within the Raw Data Table to generate second data records in a Processed Data Table that estimate the benchmark timing defined between two benchmark data records;

wherein the benchmark timing generated and stored within the processed data table is determined from difference between two data entries stored within the raw data table.